

CLAIMS

What is claimed is:

- 1 1. A method for cross-module in-lining, comprising:
 - 2 in a first phase of a compiling process,
 - 3 deciding to in-line a first function in a first module into a second
 - 4 function in a second module;
 - 5 providing the location of the first function;
 - 6 providing instructions for in-lining to be performed in a second
 - 7 phase of the compiling process;
 - 8 in the second phase of the compiling process,
 - 9 following the instructions to in-line code of the first function into
 - 10 the second function.

- 1 2. The method of claim 1 wherein:
 - 2 the compiling process comprising a front-end phase, an inter-procedural
 - 3 analysis phase, and a back-end phase;
 - 4 the inter-procedural phase being the first phase; and
 - 5 the back-end phase being the second phase.

- 1 3. The method of claim 1, in the first phase of the compiling process, further having a
 - 2 third function in the module containing the second function.

- 1 4. The method of claim 3, in the second phase of the compiling process, further
2 getting rid of the third function in the module containing the second function after
3 using that third function to in-line its code into the second function.
- 1 5. The method of claim 4 wherein the third function being selected from a group
2 consisting of the first function and a clone of the first function.
- 1 6. The method of claim 1, wherein, in the second phase of the compiling process, in-
2 lining the code of the first function into the second function uses a clone of the
3 first function.
- 1 7. The method of claim 1, wherein, in the second phase of the compiling process, the
2 code used to be in-lined into the second function is stored in a file.
- 3 8. The method of claim 1 wherein, in the second phase of the compiling process, the
4 code used to be in-lined into the second function is stored in a library.
- 1 9. The method of claim 1 wherein the instructions include at least a list of callees to
2 be in-lined and corresponding callers.
- 1 10. A method for compiling a first set of modules having programming source code,
2 comprising:
3 in a first phase,
4 from the first set of modules, providing a second set of modules
5 having first intermediate representations;
6 in a second phase,

7 performing in-line analysis on the second set of modules;
8 providing instructions for in-lining to be performed in a third phase
9 of the compiling process; and
10 providing a third set of modules having second intermediate
11 representations optimized from the first intermediate
12 representations;
13 in the third phase of the compiling process,
14 following the instructions to perform in-lining, and
15 providing a fourth set of modules having third intermediate
16 representations optimized from the second intermediate
17 representations.

1 11. The method of claim 10, in the second phase, further using code in the module
2 containing a function caller of a function callee to transform in-lining.

1 12. The method of claim 11 wherein the code being selected from a body of the
2 function callee.

1 13. The method of claim 11 wherein the code being selected from a clone of the
2 function callee.

1 14. The method of claim 10 wherein the instructions include at least one of:
2 a set of function caller including at least one function caller;
3 a set of function callee including at least one function callee;
4 the order for transformation of in-lining;

5 the location of at least one function callee; and
6 decisions whether to keep a body of at least one function callee after in-
7 lining transformation.

1 15. A computer-readable medium embodying a compiler, the compiler comprising:
1 a front-end phase;
2 a cross-module analysis phase; and
3 a back-end phase;
4 wherein
5 the front-end phase invokes the cross-module analysis phase;
6 the cross-module analysis phase
7 determines whether a callee is to be in-lined into a caller in
8 the back-end phase;
9 provides instructions for the back-end phase to transform in-
10 lining code of the callee; and
11 invokes the back-end phase; and
12 the back-end phase
13 transforms the in-lining code based on the instructions.

1 16. The computer-readable medium of claim 15 wherein the back-end phase further
2 performs tasks related to in-lining.

1 17. The computer-readable medium of claim 16 wherein the tasks related to in-lining
2 include at least deleting the callee in a module containing the caller.

1 18. The computer readable medium of claim 15 wherein transforming the in-lining
2 code uses code of a clone of the callee.

1 19. The computer-readable medium of claim 15 wherein a call to the callee is in a
2 module that does not include the callee.

1 20. The computer-readable medium of claim 15 wherein the instructions include at
2 least a list of callees.